BigData task

1. Describe at least one of data storage services – HDFS, Amazon S3, WASB (Azure blob)

2. Explain what is Apache Spark, Spark SQL (standalone mode, distributed mode).

[**About Blob storage**](#_l5fpd089qyng) **2**

[**About Azure Data Lake Storage Gen2**](#_p5q1qjvsgocq) **2**

[**Blob storage resources**](#_yer36cs9uau0) **2**

[Storage accounts](#_bj0ehqtveebk) 3

[Containers](#_1cfcfnssmoqw) 3

[Blobs](#_akk89pq4wpub) 4

[**Move data to Blob storage**](#_ympxfjhfzrru) **4**

[**Apache Spark defined**](#_74udekq3fr) **5**

[**Apache Spark architecture**](#_xb2qp5fxq9ha) **6**

[**What is Apache Spark SQL?**](#_fixqr2j0rtby) **6**

[**Spark Deploy Modes**](#_bbnn2nsvmawz) **7**

[Client Deploy Mode in Spark](#_vrpu8pf3mwh3) 7

[Cluster Deploy Mode in Spark:](#_ucswc5lozqbs) 8

Azure Blob storage is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of unstructured data. Unstructured data is data that doesn't adhere to a particular data model or definition, such as text or binary data.

## **About Blob storage**

Blob storage is designed for:

* Serving images or documents directly to a browser.
* Storing files for distributed access.
* Streaming video and audio.
* Writing to log files.
* Storing data for backup and restore, disaster recovery, and archiving.
* Storing data for analysis by an on-premises or Azure-hosted service.

Users or client applications can access objects in Blob storage via HTTP/HTTPS, from anywhere in the world. Objects in Blob storage are accessible via the Azure Storage REST API, Azure PowerShell, Azure CLI, or an Azure Storage client library. Client libraries are available for different languages, including:

* .NET
* Java
* Node.js
* Python
* Go
* PHP
* Ruby

## **About Azure Data Lake Storage Gen2**

Blob storage supports Azure Data Lake Storage Gen2, Microsoft's enterprise big data analytics solution for the cloud. Azure Data Lake Storage Gen2 offers a hierarchical file system as well as the advantages of Blob storage, including:

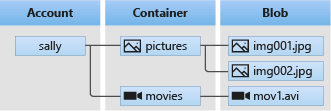
* Low-cost, tiered storage
* High availability
* Strong consistency
* Disaster recovery capabilities

## **Blob storage resources**

Blob storage offers three types of resources:

* The storage account
* A container in the storage account
* A blob in a container

The following diagram shows the relationship between these resources.



### Storage accounts

A storage account provides a unique namespace in Azure for your data. Every object that you store in Azure Storage has an address that includes your unique account name. The combination of the account name and the Blob Storage endpoint forms the base address for the objects in your storage account.

For example, if your storage account is named mystorageaccount, then the default endpoint for Blob storage is:

Copy

http://mystorageaccount.blob.core.windows.net

The following table describes the different types of storage accounts that are supported for Blob Storage:

|  |  |  |
| --- | --- | --- |
| Type of storage account | Performance tier | Usage |
| General-purpose v2 | Standard | Standard storage account type for blobs, file shares, queues, and tables. Recommended for most scenarios using Blob Storage or one of the other Azure Storage services. |
| Block blob | Premium | Premium storage account type for block blobs and append blobs. Recommended for scenarios with high transaction rates or that use smaller objects or require consistently low storage latency. Learn more about workloads for premium block blob accounts... |
| Page blob | Premium | Premium storage account type for page blobs only. Learn more about workloads for premium page blob accounts... |

### Containers

A container organizes a set of blobs, similar to a directory in a file system. A storage account can include an unlimited number of containers, and a container can store an unlimited number of blobs.

A container name must be a valid DNS name, as it forms part of the unique URI used to address the container or its blobs. Follow these rules when naming a container:

* Container names can be between 3 and 63 characters long.
* Container names must start with a letter or number, and can contain only lowercase letters, numbers, and the dash (-) character.
* Two or more consecutive dash characters aren't permitted in container names.

The URI for a container is similar to:

https://myaccount.blob.core.windows.net/mycontainer

### Blobs

Azure Storage supports three types of blobs:

* Block blobs store text and binary data. Block blobs are made up of blocks of data that can be managed individually. Block blobs can store up to about 190.7 TiB.
* Append blobs are made up of blocks like block blobs, but are optimized for append operations. Append blobs are ideal for scenarios such as logging data from virtual machines.
* Page blobs store random access files up to 8 TiB in size. Page blobs store virtual hard drive (VHD) files and serve as disks for Azure virtual machines.

The URI for a blob is similar to:

https://myaccount.blob.core.windows.net/mycontainer/myblob

or

https://myaccount.blob.core.windows.net/mycontainer/myvirtualdirectory/myblob

Follow these rules when naming a blob:

* A blob name can contain any combination of characters.
* A blob name must be at least one character long and cannot be more than 1,024 characters long, for blobs in Azure Storage.
* Blob names are case-sensitive.
* Reserved URL characters must be properly escaped.
* The number of path segments comprising the blob name cannot exceed 254. A path segment is the string between consecutive delimiter characters (e.g., the forward slash '/') that corresponds to the name of a virtual directory.

Avoid blob names that end with a dot (.), a forward slash (/), or a sequence or combination of the two. No path segments should end with a dot (.).

## **Move data to Blob storage**

A number of solutions exist for migrating existing data to Blob storage:

* AzCopy is an easy-to-use command-line tool for Windows and Linux that copies data to and from Blob storage, across containers, or across storage accounts. For more information about AzCopy, see Transfer data with the AzCopy v10.
* The Azure Storage Data Movement library is a .NET library for moving data between Azure Storage services. The AzCopy utility is built with the Data Movement library. For more information, see the reference documentation for the Data Movement library.
* Azure Data Factory supports copying data to and from Blob storage by using the account key, a shared access signature, a service principal, or managed identities for Azure resources. For more information, see Copy data to or from Azure Blob storage by using Azure Data Factory.
* Blobfuse is a virtual file system driver for Azure Blob storage. You can use blobfuse to access your existing block blob data in your Storage account through the Linux file system. For more information, see How to mount Blob storage as a file system with blobfuse.
* Azure Data Box service is available to transfer on-premises data to Blob storage when large datasets or network constraints make uploading data over the wire unrealistic. Depending on your data size, you can request Azure Data Box Disk, Azure Data Box, or Azure Data Box Heavy devices from Microsoft. You can then copy your data to those devices and ship them back to Microsoft to be uploaded into Blob storage.
* The Azure Import/Export service provides a way to import or export large amounts of data to and from your storage account using hard drives that you provide. For more information, see Use the Microsoft Azure Import/Export service to transfer data to Blob storage.

## **Apache Spark defined**

Apache Spark is a data processing framework that can quickly perform processing tasks on very large data sets, and can also distribute data processing tasks across multiple computers, either on its own or in tandem with other distributed computing tools. These two qualities are key to the worlds of big data and machine learning, which require the marshalling of massive computing power to crunch through large data stores. Spark also takes some of the programming burdens of these tasks off the shoulders of developers with an easy-to-use API that abstracts away much of the grunt work of distributed computing and big data processing.



1 second of 27 secondsVolume 0%

From its humble beginnings in the AMPLab at U.C. Berkeley in 2009, Apache Spark has become one of the key big data distributed processing frameworks in the world. Spark can be deployed in a variety of ways, provides native bindings for the Java, Scala, Python, and R programming languages, and supports SQL, streaming data, machine learning, and graph processing. You’ll find it used by banks, telecommunications companies, games companies, governments, and all of the major tech giants such as Apple, Facebook, IBM, and Microsoft.

## **Apache Spark architecture**

At a fundamental level, an Apache Spark application consists of two main components: a driver, which converts the user's code into multiple tasks that can be distributed across worker nodes, and executors, which run on those nodes and execute the tasks assigned to them. Some form of cluster manager is necessary to mediate between the two.

Out of the box, Spark can run in a standalone cluster mode that simply requires the Apache Spark framework and a JVM on each machine in your cluster. However, it’s more likely you’ll want to take advantage of a more robust resource or cluster management system to take care of allocating workers on demand for you. In the enterprise, this will normally mean running on Hadoop YARN (this is how the Cloudera and Hortonworks distributions run Spark jobs), but Apache Spark can also run on Apache Mesos, Kubernetes, and Docker Swarm.

If you seek a managed solution, then Apache Spark can be found as part of Amazon EMR, Google Cloud Dataproc, and Microsoft Azure HDInsight. Databricks, the company that employs the founders of Apache Spark, also offers the Databricks Unified Analytics Platform, which is a comprehensive managed service that offers Apache Spark clusters, streaming support, integrated web-based notebook development, and optimized cloud I/O performance over a standard Apache Spark distribution.

Apache Spark builds the user’s data processing commands into a Directed Acyclic Graph, or DAG. The DAG is Apache Spark’s scheduling layer; it determines what tasks are executed on what nodes and in what sequence.

## **What is Apache Spark SQL?**

Spark SQL brings native support for SQL to Spark and streamlines the process of querying data stored both in RDDs (Spark’s distributed datasets) and in external sources. Spark SQL conveniently blurs the lines between RDDs and relational tables. Unifying these powerful abstractions makes it easy for developers to intermix SQL commands querying external data with complex analytics, all within in a single application. Concretely, Spark SQL will allow developers to:

* Import relational data from Parquet files and Hive tables
* Run SQL queries over imported data and existing RDDs
* Easily write RDDs out to Hive tables or Parquet files

Spark SQL also includes a cost-based optimizer, columnar storage, and code generation to make queries fast. At the same time, it scales to thousands of nodes and multi-hour queries using the Spark engine, which provides full mid-query fault tolerance, without having to worry about using a different engine for historical data.

Spark deployment mode (--deploy-mode) specifies where to run the driver program of your Spark application/job, Spark provides two deployment modes, client and cluster, you could use these to run Java, Scala, and PySpark applications.

Using <strong>spark-submit --deploy-mode <client/cluster></strong> , you can specify where to run the Spark application driver program.

## **Spark Deploy Modes**

|  |  |
| --- | --- |
| VALUE | DESCRIPTION |
| cluster | In cluster mode, the driver runs on one of the worker nodes, and this node shows as a driver on the Spark Web UI of your application. cluster mode is used to run production jobs. |
| client | In client mode, the driver runs locally from where you are submitting your application using spark-submit command. client mode is majorly used for interactive and debugging purposes. Note that in client mode only the driver runs locally and all tasks run on cluster worker nodes. |

If you wanted to know the deploy mode of running or completed Spark application, you can get it by accessing Spark Web UI from Spark History Server UI and check for spark.submit.deployMode property on Environment tab

### Client Deploy Mode in Spark

In client mode, the Spark driver component of the spark application will run on the machine from where the job submitted.

In a typical Cloudera cluster, you submit the Spark application from the Edge node hence the Spark driver will run on an edge node.

In a Spark Standalone Cluster, the driver runs on a master node (dedicated server) with dedicated resources.

spark-submit --deploy-mode client --driver-memory xxxx ......

Copy

* The default deployment mode is client mode.
* In client mode, if a machine or a user session running spark-submit terminates, your application also terminates with status fail.
* Using Ctrl-c after submitting the spark-submit command also terminates your application.
* Client mode is not used for Production jobs. This is used for testing purposes.
* Driver logs are accessible from the local machine itself.

Note: Network Overhead – As data needs to be moved between the driver and the worker nodes across the network (between the submitting machine(driver machine) and the cluster), depending on the network latency you may notice performance degradation.

### Cluster Deploy Mode in Spark:

In Cluster Deploy mode, the driver program would be launched on any one of the spark cluster nodes (on any of the available nodes in the cluster). Cluster deployment is mostly used for large data sets where the job takes few mins/hrs to complete.

spark-submit --deploy-mode cluster --driver-memory xxxx ........

Copy

* Terminating the current session doesn’t terminate the application. The application would be running on the cluster. You can get the status of the spark application by running <strong>spark-submit --status [submission ID]</strong>
* Since Spark driver runs on one of the worker node within the cluster, which reduces the data movement overhead between submitting machine and the cluster.
* For the Cloudera cluster, you should use yarn commands to access driver logs.
* In this spark mode, the change of network disconnection between driver and spark infrastructure reduces. As they reside in the same infrastructure(cluster), It highly reduces the chance of job failure.